



Zaidi, SIH., Nabina, A., Canagarajah, CN., & Nunez-Yanez, JL. (2008). Power/area analysis of a FPGA-based open-source processor using partial dynamic reconfiguration. In *EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, 2008 (DSD '08), Parma* (pp. 592 - 598). Institute of Electrical and Electronics Engineers (IEEE). <https://doi.org/10.1109/DSD.2008.92>

Peer reviewed version

Link to published version (if available):  
[10.1109/DSD.2008.92](https://doi.org/10.1109/DSD.2008.92)

[Link to publication record in Explore Bristol Research](#)  
PDF-document

## University of Bristol - Explore Bristol Research

### General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:  
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/>

## POWER/AREA ANALYSIS OF A FPGA-BASED OPEN-SOURCE PROCESSOR USING PARTIAL DYNAMIC RECONFIGURATION

*Izhar Zaidi, Atukem Nabina, Professor CN Canagarajah, Jose Nunez-Yanez*

Department of Electrical and Electronic Engineering  
University of Bristol, UK

Email: [izhar.zaidi@bris.ac.uk](mailto:izhar.zaidi@bris.ac.uk), [Atukem.Nabina@bristol.ac.uk](mailto:Atukem.Nabina@bristol.ac.uk), [Nishan.Canagarajah@bristol.ac.uk](mailto:Nishan.Canagarajah@bristol.ac.uk),  
[j.l.nunez-yanez@bristol.ac.uk](mailto:j.l.nunez-yanez@bristol.ac.uk)

### ABSTRACT

This paper explores the utilization of run-time Partial Dynamic Reconfiguration in the LEON3 open-source soft core processor, which is a highly configurable SPARC (Scalable Processor ARChitecture) V8 instruction set processor. The work explores the possibilities of sharing different arithmetic functions tightly coupled to the integer pipeline and mapped to the same silicon area, saving power consumption and area utilisation. The same strategy can be used to extend the instruction set architecture of the processor with new instructions that are optimized for DSP applications. The logic necessary to support these instructions could then be swapped as demanded by the application.

### 1. INTRODUCTION

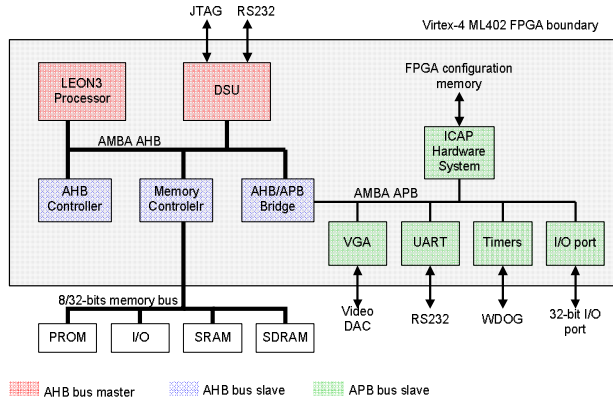
The increase in transistor density in FPGAs has enabled the integration of complex systems in a single fabric at the expense of high device costs and power consumption. The introduction of Partial Dynamic Reconfiguration (PDR) in some FPGAs allows the partitioning of a complex application into modules which can be time multiplexed in a smaller device with potential savings in power and area. Partial Dynamic Reconfiguration is a technique in which part of the design is reconfigured without interrupting the rest of the system performing computation. Xilinx has made this possible with the introduction of ICAP (Internal Configuration Access Port) in the Virtex-II, IV and V families enabling the user to modify the circuit without an external controller. Traditionally, the configuration of a Xilinx FPGA is controlled externally by writing byte-wide data to the configuration memory through the selectMAP utility. The ICAP, on the other hand, allows parallel reading and writing of the configuration memory internally i.e. from inside the FPGA.

A system without Partial Dynamic Reconfiguration requires a configuration design fixed at compilation time. In order to change the functionality of the system, its

operation must be interrupted or terminated and an external controller is required to load the new design. In recent years the requirement of flexibility in architecture demands the system to be able to cope with changes in protocol structure, data coding standards, enhanced algorithms and functionalities etc drifting towards the use of adaptive and reconfigurable hardware.

Partial Dynamic Reconfiguration has been an extensive topic of investigation in the last decade with applications ranging from Satellite communication systems [1] and Erlangen Slot Machine [2] to adaptive control system [3], [4]. The advantages gained from reconfiguration have led to the research and development of a virtual internal configuration access port (JCAP) [5] in a Xilinx Spartan III platform, which does not support internal re-configuration by default. For Self-Reconfiguration, it is possible to make use of an internal processor [1], [4], [5], [6] or a dedicated controller [2], [3], which resides inside the fabric in order to reprogram part of the system without interrupting the current running processes. The internal processors that have been extensively used to control Partial Dynamic Reconfiguration in the Xilinx family of devices have been the IBM PowerPC hard IP core and the MicroBlaze soft core processor. MicroBlaze is a 32-bit, 3 stage integer pipeline reduced instruction set computer (RISC) optimized for implementation in Xilinx FPGAs. In addition to its fixed features, the MicroBlaze processor is parameterised to allow selective enabling of additional functionality. The multiplier for the MicroBlaze processor has a latency of 3 clock cycles when implemented in hardware and the divider has a latency of 34 clock cycles. MicroBlaze is distributed with the Xilinx Embedded Development Kit (EDK) as an encrypted netlist and the source code can be obtained from Xilinx at a higher cost.

For this work, we have selected LEON3 over MicroBlaze core due to its open source nature and its flexibility, which enables the modification of the internal micro-architecture of the core. The experimental setup makes use of the integer pipeline based hardware multiplier and divider module from LEON3 core and utilises them as per need basis with the help of Partial Dynamic Reconfiguration to



**Fig. 1.** LEON3 System and subsystems.

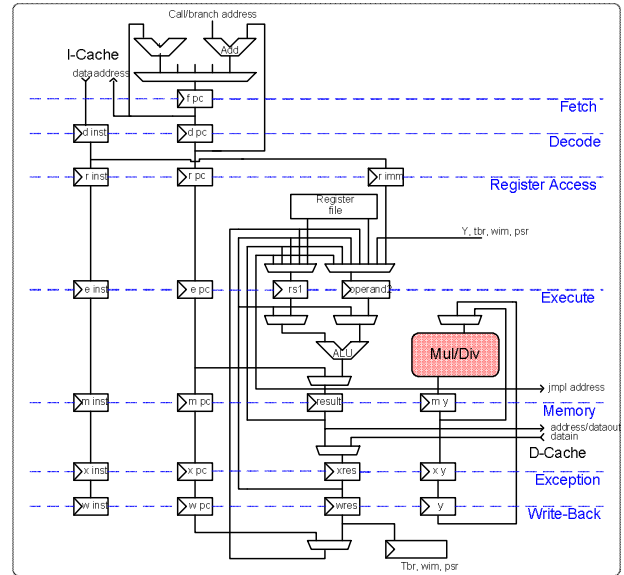
save area and power consumption. The paper is organised as follows. In Section 2, we will describe the overall system that allows Partial Dynamic Reconfiguration using LEON3. Section 3 will discuss the implementation and the related problems. Section 4 will discuss the results and finally section 5 will discuss the future work.

## 2. DESCRIPTION OF SYSTEM

### 2.1 LEON3 System

LEON3 micro architecture is implemented using a 7-stage pipeline with separate instruction and data cache buses. It supports the full SPARC V8 instruction set [7]. The LEON3 processor is described in synthesizable VHDL and is available from Gaisler Research under the GNU license. The benefit of having it as a synthesizable VHDL is that it can be configured modified and adapted for specific applications. It is a bus centric system using the AMBA-2.0 AHB/APB allowing most of the modules to interact through AMBA. Plug & Play functionality is an included feature of the LEON3 system therefore allowing mapping of peripherals to be implemented in software. The LEON3 system is provided with a number of generic modules which can be used to test and debug the entire system. The LEON3 system is shown in Fig. 1.

AMBA is the default bus system that comes with the LEON3 system used to interconnect modules as show in Fig. 1. There are 3 types of buses in the AMBA bus hierarchy; AHB (Advanced High-performance Bus), ASB (Advanced System Bus) and APB (Advanced Peripheral Bus). Within the LEON3 soft-core processor only 2 out of the 3 bus types are used; AHB and APB. AHB is mainly used for interconnecting bus masters and slaves that require some or most of the complexity of the AHB bus interface system. Such complexities can be split transaction capabilities, bus transfer definition and specific

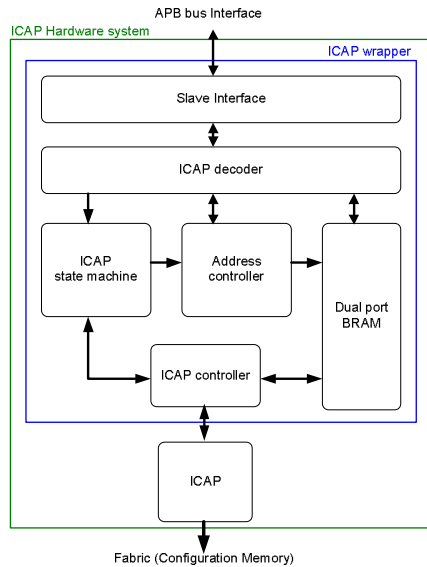


**Fig. 2.** Mul/Div module in integer unit pipeline

transfer acknowledgements. The APB bus however is a subset of the AHB and provides a less complex mechanism of interaction between AHB masters and the slaves on the APB bus. It is optimised for reduced consumption in power and minimal interfacing complexity. It is mainly designed for interfacing peripherals with smaller address spaces [8].

### 2.2 Reconfigurable modules and systems

The Mul/Div unit is used to test the instruction set extendibility of the micro architecture of the integer unit of LEON3 as illustrated in Fig. 2. It must be enabled to conform to the SPARC V8 specification. There are different types of multipliers modules that can be selected to be used within the LEON3 integer unit and these modules are differentiated by latency, size of operands and total logic area required for implementation. An example of this is the 32x16 bit multiplier with 2 clock cycles of latency and the 16x16 bit multiplier with 4 clock cycles of latency. The implementation of the Mul/Div unit is made configurable using Partial Dynamic Reconfiguration allowing the choice of multiplier modules or divider module to be selectable at runtime in order to maximise efficiency of these operations in terms of latency, power consumption and area utilisation. The multiplier module selected for this work implements 32x32 bit multiplication. It takes two signed or unsigned numbers as input and produces a 64-bit result. The latency of this multiplier is one clock cycle [9]. The selected divider module performs signed/unsigned 64-bit by 32-bit division taking 36 clock cycles and leaving no remainder. It implements the radix-2 non-restoring iterative division algorithm [9]. One of these



**Fig. 3.** ICAP Hardware modular systems.

2 units is instantiated into the reconfigurable region of the FPGA at runtime as required by the user application

ICAP is the primitive used to provide access to the fabric of the FPGA and hence the configuration memory of the device. It provides functionalities such as the ability to read and write to its internal registers and also to read and write to the configuration memory. The ICAP wrapper, as shown in Fig. 3, is the module containing the hardware controllers used to provide smooth interaction and useful functionalities between the software and the ICAP.

The ICAP decoder module is the main controller of the ICAP wrapper system. It filters the data received from the APB Bus and differentiates between instructions and data that is to be used for Partial Dynamic Reconfiguration and asserts the right control signals to the rest of the modules in the ICAP wrapper system. The Address controller and ICAP controller are the modules used to control the flow of data within the ICAP wrapper as data can both be sent to the ICAP primitive in order to implement reconfiguration of an area of the FPGA or read back from ICAP. The read back data would represent the particular implementation of a region of the FPGA stored in the configuration memory of the device, or data from the internal registers of ICAP. The Dual Port BRAM is used as a buffer memory. Data to be written into ICAP and data read back from ICAP is temporally stored into the BRAM until it is used. Both the ICAP wrapper and ICAP are contained within a module known as the ICAP hardware system which is connected as a slave to APB.

## 2.3 Platform

The ML402 Virtex-4 VSX 35 device from Xilinx family of devices is the FPGA platform used for implementation of the LEON3 soft-core processor and the ICAP hardware system. It is configured with the LEON3 system by loading its configuration memory with the bitstream that contains the implementation information. Partial Dynamic Reconfiguration of the platform is implemented by only modifying the frame regions of the FPGAs configuration memory to which the partial bitstream refers to.

The configuration memory of Virtex-4 family of devices is composed of frame regions which define the smallest granules of fabric that can be reconfigured. When a portion of a frame needs to be reconfigured, the partial bitstream still contains the full frame information, including the portion of the frame that is not modified. The ML402 contains 10410 configurable frames 660 non configurable frames therefore a total of 11070 with each frame having 41 words (4 bytes). However it is important to note that only configurable frames count towards the overall size of the bitstream used for full and Partial Dynamic Reconfiguration of the device [10].

## 2.4 PlanAhead

PlanAhead is an advance hierarchical floorplanning tool provided by Xilinx to facilitate the design and analysis of large FPGA devices and Partial Dynamic Reconfigurable systems. This tool can partition the physical design into smaller designs reducing the time to understand, verify, and implement it on to the target FPGA. The software allows the import of a netlist describing the system, floorplanning for both static and partial reconfiguration systems, design rule checks, timing analysis for different implementations, bitstream generation, resource utilisation, interconnect delay and routing connectivity for FPGA configuration. As the design can be partitioned into physical blocks known as PBlocks it allows constraining of modules e.g. the reconfigurable modules, to a particular area of the FPGA. If PBlocks are not constrained to a particular region, their distribution on the floorplan is left to the software allowing the unconstrained design to be grouped during placement. In partial reconfiguration design it is necessary to assign a PBlock to both static and reconfigurable module and to constrain the reconfigurable module PBlock to an area on the FPGA. This area is known as the reconfigurable region and bus macros are site constrained to interface signals between this region and the static design.



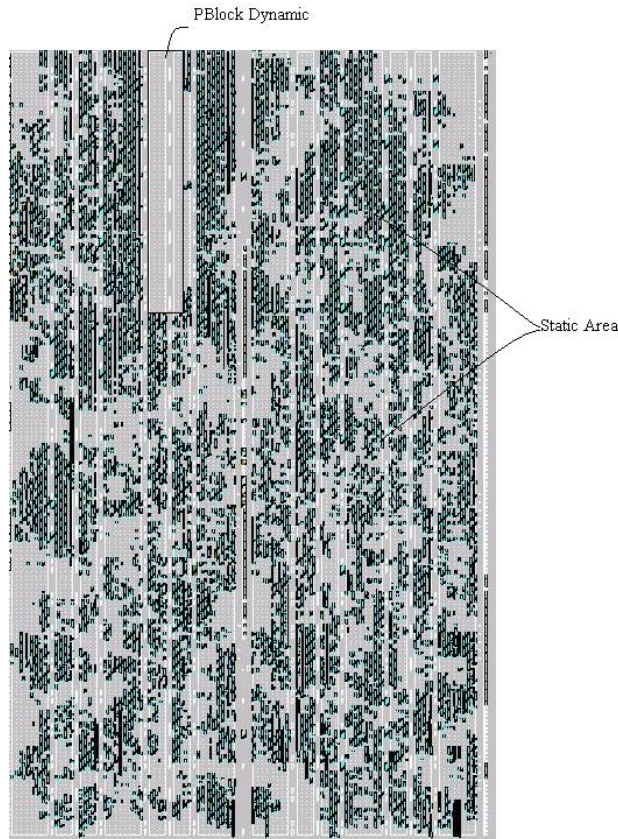


Fig. 4. Floorplan of LEON3 with Dynamic PBlock

### 3. IMPLEMENTATION

To build an experimental setup for partial reconfiguration, the hardware multiplier and divider structure from the LEON3 core was moved to the top-level hierarchy of the system. In order to do that the guidelines for partial reconfiguration were followed and a new top-level file was introduced which contains the static design, which was LEON3 itself, and a dynamic design containing either multiplier or a divider [11]. The pads from the previous LEON3 top-file were also moved and connected on the new top-level, as pads are not allowed in static design according to the architecture flow of Xilinx tools.

For self-reconfiguration, the ICAP hardware source code provided by Xilinx were modified and attached to LEON3 APB bus. The source code is written and supported for Xilinx PowerPC 405, MicroBlaze processor and OPB bus architecture and needs to be tailored accordingly for the AMBA bus architecture. The work started with introducing some address space for the ICAP as a slave on the APB bus, the wrapper was created to bridge the signals of the AMBA bus to the OPB bus inside the ICAP source code. The ICAP hardware system requires 4 clock cycles to

Table 1. Source Meter Properties

	Source range		Measurement range	
	Lower	Upper	Lower	Upper
Voltage	5 $\mu$ V	63V	1 $\mu$ V	63.3V
Current	500pA	3.15A	100pA	3.165A
Resistance	-	-	10 $\mu\Omega$	21.1M $\Omega$

complete a read/write cycle so the delay was introduced in APB to wait for the ICAP hardware system to finish its task. Some problem were encountered while trying to match the AMBA cycle with the ICAP hardware system and also to constraint the ICAP itself as Virtex-4 has two ICAP sites top and bottom. The top site should be location constraint otherwise the tools will automatically place it in the bottom site and will not work according to answer record from the early access lounge on Xilinx website [12]. To allow communication between the processor and the partial module, bus macros are used, as this is the limitation of the tools. Bus macros are instantiated as black boxes and are programmed with a predefined routing macro.

The static design and the dynamic multiplier and divider module were synthesized separately producing a netlist. The modules were synthesized without the option of inserting pads and the top level file was synthesized with the insert pads option. These netlist files were then used as inputs into the Xilinx PlanAhead software to produce a partial reconfigurable design. In order to produce separate bitstreams for both multiplier and divider, the multiplier was selected first in the PlanAhead project along with the top-level and static design. Then a separate PBlock for static and dynamic areas were created, and an area was defined where the dynamic block was planned to be placed as shown in Fig. 4. The ICAP was then location constrained and finally the bus macros were placed. After the completion of initial setup outlined above, the static design was implemented (placed and routed) and then the dynamic part was implemented. After the design was placed & routed then the last step assemble process was performed. The assemble produces three bitstreams, first is static-full bitstream which contains the complete design which is a combination of static design and one of the reconfigurable modules as the default start up implementation, then dynamic bitstream and the last is the blank bitstream. For generating the bitstreams with divider module, the net list is updated replacing multiplier with the divider and re-running the dynamic implementation and bitstream generation.

**Table 2.** Resource Utilisation and power analysis

	Test case 1		Test case 2	Test case 3
Number of slices utilised	8883		8883 + 452	8883 + 452
Size of bitstream	1673		1673 + 28	1673 + 26
% FPGA Utilisation	57		57 + 2	57 + 2
Static Power Consumption (mW)	624		627	627
Total Power consumption at runtime (mW)	795	760	685	653

## 4. IMPLEMENTATION RESULTS

### 4.1 Test environment

The ML402 board was modified to allow the FPGA core voltage to be user controlled and to allow the measurement of voltage and current. This modification only allows the control of the internal core voltage and does not affect the I/O port voltage. This was achieved by disconnecting the voltage regulator providing power to the core of the FPGA and making the voltage terminals of the FPGA core accessible through unused pins on the GPIO port of the board. The Keithley 2420 source meter was then used to provide voltage to the FPGA internal core and current and time information was measured to obtain the power used by the internal core of the device. The source meter characteristics are shown in Table 1.

### 4.2 The Keithley control program

A program referred to as the “Keithley source meter control program” was written to allow control of the settings of the Keithley source meter as well as data extraction from the source meter to a host computer. Useful parameters such as voltage, number of samples and sampling period was configured with the help of the program and data was extracted and returned to the program as defined by the parameters. The data that was extracted was voltage, current and time interval between measurements. This information was automatically stored in a spreadsheet. From these results power and energy information was computed.

**Table 3.** Power and Energy Analysis of Reconfiguration

	Clock cycles to PDR	Average Power during PDR (mW)	PDR duration (mSec)	PDR Energy (mJ)
Mul	821440	730	16.4	12
Div	780544	730	15.5	11.3

### 4.3 Test conditions

As outlined in Table 2, there were 3 test cases that were used for testing. Test case 1 was the complete LEON3 system with the ICAP hardware system and a partial reconfiguration region. However the reconfigurable region which is the Mul/Div unit in the integer pipeline contained no module. Test case 2 was a replica of Test case 1 with the multiplier instantiated in the configurable region. Test case 3 was also a replica of Test case 1 with the divider instead of the multiplier.

The Keithley source meter control program, with the aid of a computer was then used to control the voltage applied to the core of the FPGA from the Keithley source meter with each of the test cases being evaluated. Enough time was allowed for the current to stabilise then 500 sequential measurements of voltage, current and time were taken for each case. The voltage and current information was averaged and used to calculate the static power consumption of the device as presented in Table 2. row titled Static Power consumption. For these tests, no application program was run on the LEON3 processor and the operational frequency of the system for all test cases was 50 MHz.

### 4.4 LEON3 program application

Another test was also carried out in order to measure power consumption of the system at runtime and the duration of partial reconfiguration. This test was carried out by designing an application program to be run on the LEON3 processor. The bitstreams used for partial reconfiguration were appended to the application program and the ensemble was loaded into the DDR memory of the LEON3 system where it could be accessed and used as required.

The application program starts with the initialisation process of ICAP and the timer module. The initialisation process consisted of read/write sequences of data transfers to the control registers and status registers of the ICAP hardware system, ICAP, and the timer module in order to configure these modules for use. Then, assuming that the reconfigurable module is initially blank, the application starts the timer and runs the partial reconfiguration

function which transfers the divider bitstream from the memory to the ICAP hardware system. The presence of the divider is then validated by running a division instruction in assembly language and the number of clock cycles that was taken for the partial reconfiguration to be completed is noted. In the next step the timer is reset and the partial reconfiguration function for the multiplier is executed. The number of clock cycles taken for the partial reconfiguration is noted and the structure is validated by performing a multiplication in assembly language. The power consumption of the LEON3 system during reconfiguration is estimated by implementing a repetitive reconfiguration of either the multiplier or the divider in the application program and by taking 500 readings during this process. These 500 readings are then analysed and averaged in order to obtain an estimate of the power consumption of the system. It is important to note that the power level of the system during the repetitive reconfiguration process gradually increases until it reaches a point of stability. It is at this point that the 500 sequential measurements are taken and averaged to estimate the power consumption of the reconfiguration process. This method of power measurement provides the worst case values for this particular implementation. Therefore, at runtime, the power consumption would be less than the measurements outlined in by the results.

#### 4.5 Analysis of results

Table 2 and 3 depicts the result from the experiments. Table 2 shows that 57% of slices are occupied by the static module and 2% of the area is utilised by dynamic multiplier/divider which supports our initial statement of saving area utilisation by time multiplexing modules. Table 2 also shows the power estimate of our experiment, the static design with no module in the reconfiguration region dissipates about 624mW without any program running and compared to one with multiplier or divider module instantiated, it saves approximately 3mW. The run-time power readings for soft multiplication and division were also noted in table 2 test case 1 for comparing the difference between the power saving by having a hardware multiplier or a divider unit instead of having a software algorithm implemented.

Table-3 shows the number of clock cycles required to reconfigure a specific module. As an example the reconfiguration of the divider module consumes an average power of 730mW and takes approximately 15.5mSec to reprogram a 26kb of bitstream which comes out to be a total of 11.3mJ.

## 5. CONCLUSION AND FUTURE WORK:

In this paper we have shown the feasibility of using open-source LEON3 processor with Partial Dynamic Reconfiguration in Xilinx FPGAs. We have also shown that power can be saved whilst using Partial Dynamic Reconfiguration by replacing a blank bitstream with an idle module. There is a trade-off between the time and energy required for reconfiguring that module and how often it is used as compared to the energy utilised using software algorithm. The results presented can be used as a baseline while analysing a specific application in order to determine the best approach to obtain a power and area efficient system.

The future work will include further investigation of power saving using a procedure like dynamic voltage scaling [13] along with partial reconfiguration and the feasibility of configuration overhead for significantly large reconfigurable modules. The other area to explore is the possibility of having an extended set of instructions which load the required hardware modules dynamically.

## REFERENCES

- [1] Xiaofeng Wu and Tanya Vladimirova "A Self-reconfigurable System-on-Chip Architecture for Satellite On-Board Computer Maintenance" C. Jesshope and C. Egan (Eds.): ACSAC 2006, LNCS 4186, pp. 552 – 558, 2006. Springer-Verlag Berlin Heidelberg 2006.
- [2] Mateusz Majer "An FPGA based Dynamically Reconfigurable Platform: From Concept to Realization" 10.1109/FPL.2006.311364.
- [3] Stehhen Toscher, Thomas Reinemaan, Roland Kasper "An adaptive FPGA-Based Mechatronic Control System Supporting Partial Reconfiguration of Controller Functionalities" First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06) 0-7695-2614-4/06 .
- [4] Michael Hubnre, Christian Schck, Matthias Kuhnle, Jurgen Becker "New 2-Dimensional Partial Dynamic Reconfiguration Techniques for Real-time adaptive Microelectronic Circuits" Emerging VLSI Technologies and Architectures (ISVLSI'06) 0-7695-2533-4/06.
- [5] K.Paulsson, M.Hubner, G. Auer, M. Dreschmann, L. Chen, J. Becker "Implementation of a Virtual Internal Configuration Access Port (ICAP) for Enabling Partial Self-Reconfiguration on Xilinx Spartan III FPGA's" 10.1109/FPL.2007.4380671.
- [6] P. Sedcole, B. Blodget, T.Becker, J. Anderson and P. Lysaght "Modular Dynamic Reconfiguration in Virtex FPGAs" IEE Proc.-Comput. Digit. Tech. Vol 153, No. 3, May 2006.
- [7] GAISLER RESEARCH GR LEON3 processor description.
- [8] AMBA Specification (Rev 2.0), 13th May 1999

- [9] GRLIB IP Library User's Manual Version 1.0.14 Jiri Gaisler, Sandi Habinc, Edvin Catovic, Gaisler Research, 2006.
- [10] Xilinx; "Virtex-4 Configuration Guide, ug071" January 12, 2007..
- [11] PlanAhead as Platform for Partial Reconfiguration [http://www.xilinx.com/publications/xcellonline/xcell\\_55/xc\\_pdf/xc\\_prmethod55.pdf](http://www.xilinx.com/publications/xcellonline/xcell_55/xc_pdf/xc_prmethod55.pdf)
- [12] AR#25018 - Partial Reconfiguration - PlanAhead Flow FAQ / Know Issues for the Early Access Partial PlanAhead Program <http://www.xilinx.com/support/answers/25018.htm>
- [13] Jose Luis Nunez-Yanez, Vassilios Chouliaras. Jiri Gaisler "Dynamic Voltage Scaling in a FPGA-Based System-On-Chip" 10.1109/FPL.2007.4380689.